Wine Quality Prediction Using Machine Learning

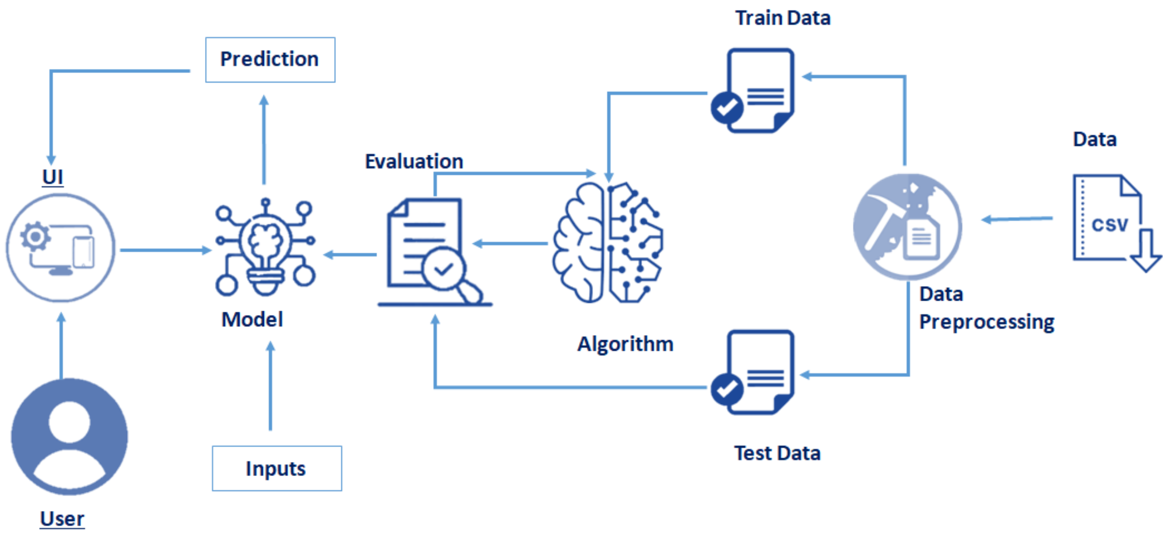
Project Description:

Wine is the most commonly used beverage globally, and its values are considered important in society. Wine is an alcoholic drink that is made up of fermented grapes. Quality of wine is important for its consumers, mainly for producers in the present competitive market to raise the revenue. Wine quality refers to the factors that go into producing a wine, as well as the indicators or characteristics that tell you if the wine is of high quality. Historically, wine quality used to be determined by testing at the end of the production.

If you have come across wine then you will notice that wine has also their type, they are red and white wine. According to experts, wine is differentiated according to its smell, flavour, and colour, but we are not wine experts to say that wine is good or bad. Every person has their own opinion about the tastes, so identifying a quality based on a person's taste is challenging. Judging the quality of wine manually is a really tough task, even the professional wine tasters have the accuracy of 71%.

In this project, we present a wine quality prediction technique that utilizes historical data to train simple machine learning models which are more accurate and can help us know the quality of wine. The models can be run on much less resource intensive environments. From this the best model is selected and saved in pkl format. We will be doing flask integration and IBM deployment.

Technical Architecture:



Project Objectives

By the end of this project:

- You'll be able to understand the problem to classify if it is a regression or a classification kind of problem.
- You will be able to know how to pre-process/clean the data using different data pre-processing techniques.
- You will be able to analyze or get insights into data through visualization.
- Applying different algorithms according to the dataset and based on visualization.
- You will be able to know how to find the accuracy of the model.
- You will be able to know how to build a web application using the Flask framework.

Prerequisites

To complete this project, you must require following software's concepts and packages

- **Anaconda navigator:**
  - Refer to the link below to know more about how download anaconda navigator
  - https://youtu.be/5mDYijMfSzs

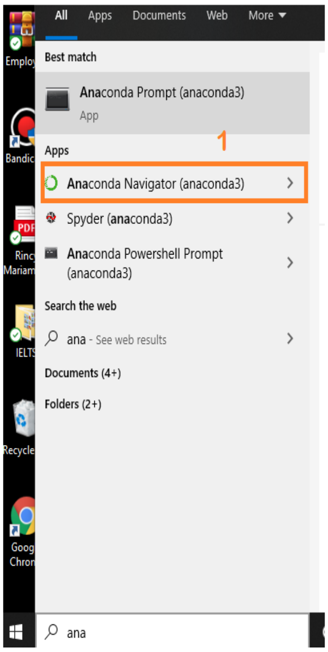
- **Python packages:**

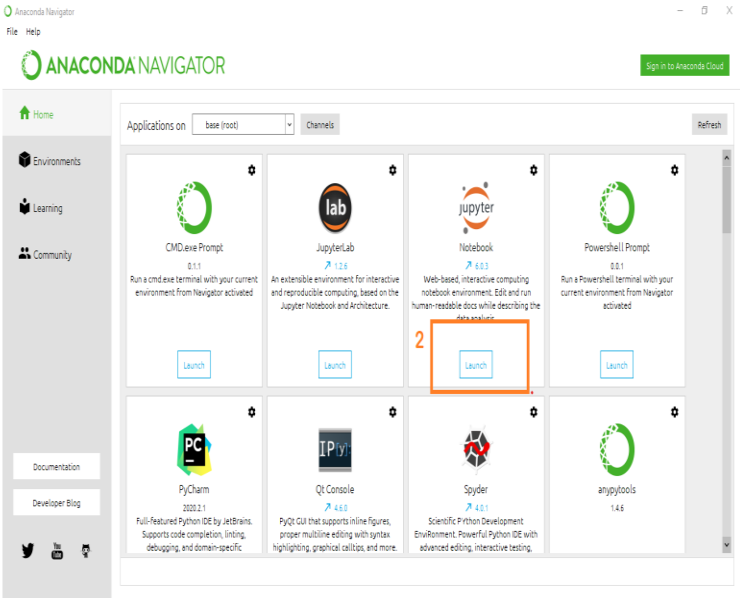
Open anaconda prompt as administrator

- Type **“pip install numpy”** and click enter.
- Type **“pip install pandas”** and click enter.
- Type **“pip install scikit-learn”** and click enter.
- Type **”pip install matplotlib”** and click enter.
- Type ”**pip install pickle-mixin”** and click enter.
- Type **”pip install seaborn”** and click enter.
- Type **“pip install Flask”** and click enter

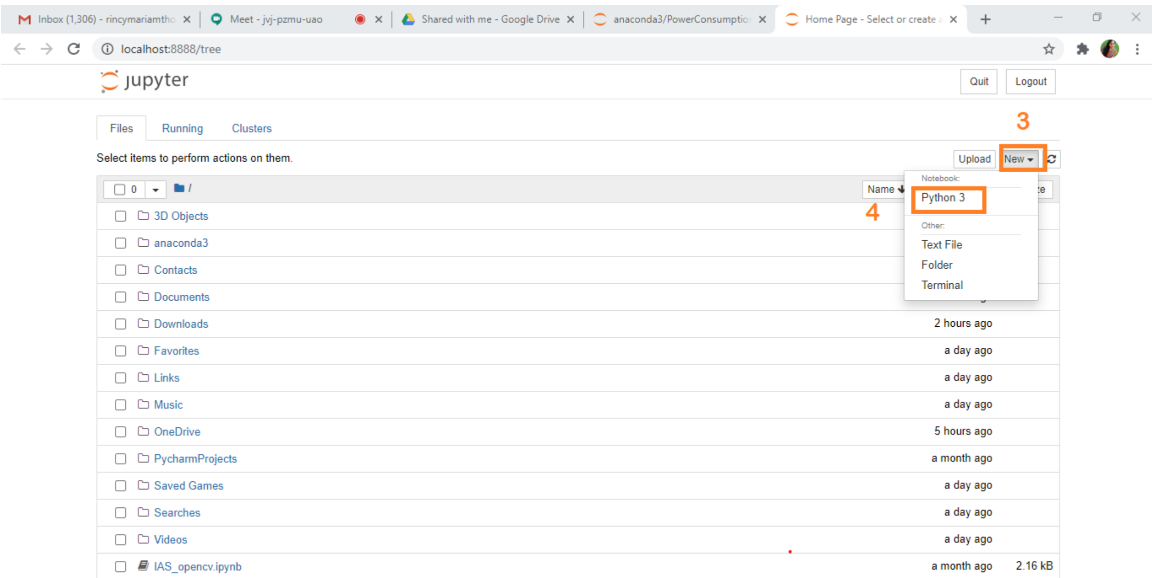
The above steps allow you to install the package in the anaconda environment

- **Launch Jupyter**
  - Search for Anaconda Navigator and open Launch Jupyter notebook.





- Then you will be able to see that the jupyter notebook runs on localhost:8888.
- To Create a new file Go to New Python3.The file in jupyter notebook is saved with .ipynb extension.



- Flask Basics : https://www.youtube.com/watch?v=lj4I_CvBnt0

Project Flow:

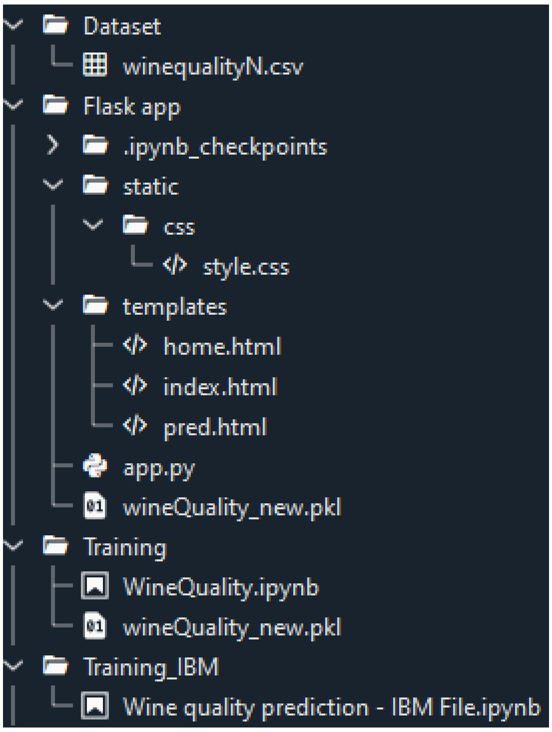
- User interacts with the UI to enter the input.
- Entered input is analyzed by the model which is integrated.
- Once model analyses the input the prediction is showcased on the UI

To accomplish this, we have to complete all the activities and tasks listed below

- Data Collection.
  - Collect the dataset or Create the dataset
- Visualizing and analyzing data
  - Univariate analysis
  - Bivariate analysis
  - Multiv nariate analysis
  - Descriptive analysis
- Data Pre-processing.
  - Drop unwanted features
  - Checking for null values
  - Handling categorical data
  - Splitting data into train and test
  - Feature Scaling
- Model Building
  - Import the model building Libraries
  - Initializing the model
  - Training and testing the model
  - Evaluation of Model
  - Save the Model
- Application Building
  - Create an HTML file
  - Build a Python Code

Project Structure

Create a Project folder that contains files as shown below



- A python file called app.py for server side scripting.
- We need the model which is saved and the saved model in this content is **wineQuality_new.pkl**
- Templates folder which contains base.HTML file, index.HTML file, prediction.HTML file.
- Static folder which contains css folder which contains style.css .

Milestone 1: Data Collection:

ML depends heavily on data, without data, it is impossible for an "AI" to learn. It is the most crucial aspect that makes algorithm training possible. In Machine Learning projects, we need a training **data set.** It is the actual **data set** used to train the model for performing various actions.

Activity1: Download the dataset

You can collect datasets from different open sources like kaggle.com, data.gov, UCI machine learning repository etc.

The dataset used for this project was obtained from Kaggle . Please refer to the link given below to download the data set and to know about the dataset

Data Pre-Processing

As we have understood how the data is. Lets pre-process the collected data.

The download data set is not suitable for training the machine learning model as it might have so much of randomness so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps.

- Removing unwanted columns
- Handling missing values
- Converting the target variable into binary class variable
- Handling categorical data
- Splitting dataset into training and test set
- Scaling Techniques

Importing The Libraries

Import the necessary libraries as shown in the image.

To know about the packages refer the link given in pre requisites.

```python
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import GridSearchCV, cross_val_score
from sklearn.svm import SVC
import pickle
```

Python

Read The Dataset

Our dataset format might be in .csv, excel files, .txt, .json, etc. We can read the dataset with the help of pandas.

In pandas we have a function called **read_csv()** to read the dataset. As a parameter we have to give the directory of csv file.

Importing the Dataset

```
data = pd.read_csv("winequalityN.csv")
data.head()
```

[2] Python

	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	white	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45	8.8	6
1	white	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	6
2	white	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6
3	white	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6
4	white	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6

Let’s know more about our data

```
data.columns
```

[3] Python

```
Index(['type', 'fixed acidity', 'volatile acidity', 'citric acid',
       'residual sugar', 'chlorides', 'free sulfur dioxide',
       'total sulfur dioxide', 'density', 'pH', 'sulphates', 'alcohol',
       'quality'],
      dtype='object')
```

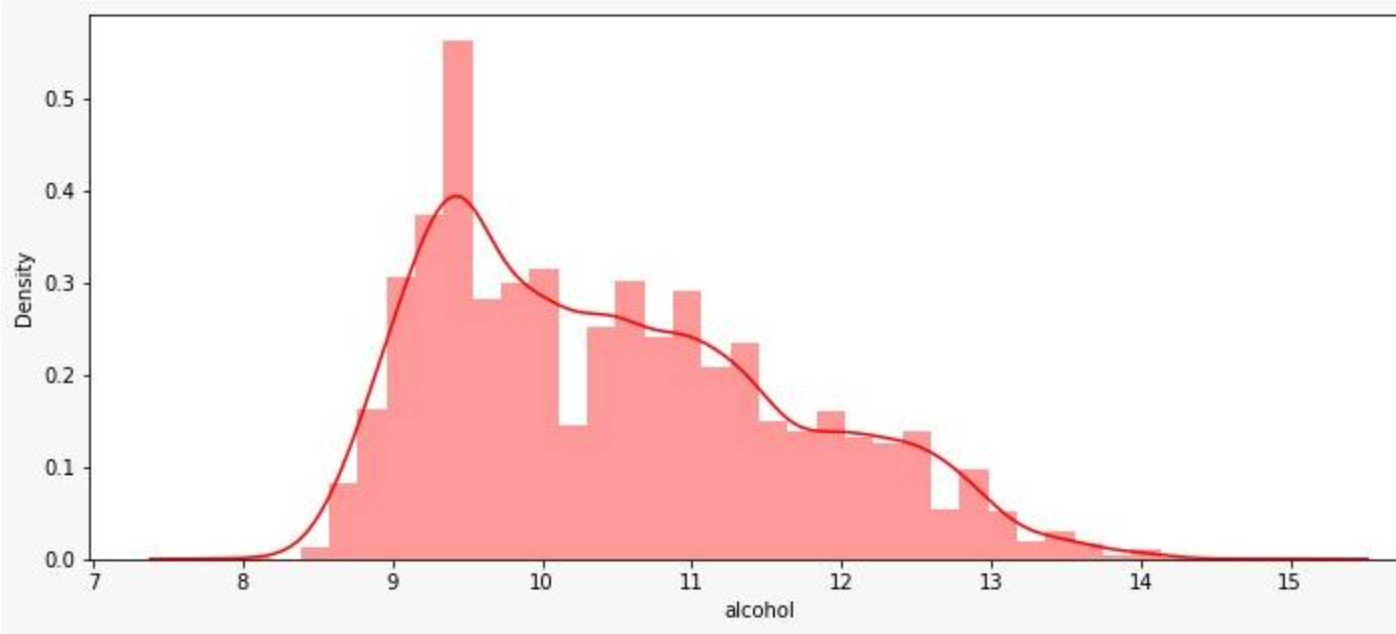
These are the different columns present in our dataset, let’s understand what these features indicates:

- volatile acidity: Volatile acidity is the gaseous acids present in wine.
- fixed acidity: Primary fixed acids found in wine are tartaric, succinic, citric, and malic
- residual sugar: Amount of sugar left after fermentation.
- citric acid: It is weak organic acid, found in citrus fruits naturally.
- chlorides: Amount of salt present in wine.
- free sulfur dioxide: So2 is used for prevention of wine by oxidation and microbial spoilage.
- total sulfur dioxide: Total SO2 used
- pH: In wine pH is used for checking acidity
- density: Density of the wine
- sulphates: Added sulphites preserve freshness and protect wine from oxidation, and bacteria.
- alcohol: Percent of alcohol present in wine.

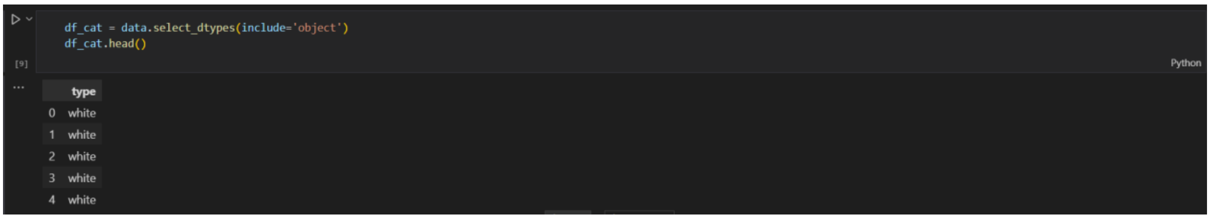
Univariate Analysis

In simple words, univariate analysis is understanding the data with single feature. Here we have displayed two different graphs such as pie plot, box plot and count plot.

- Seaborn package provides a wonderful function distplot. The distplot represents the univariate distribution of data i.e. data distribution of a variable against the density distribution. With the help of distplot, we can find the distribution of the feature. We have used distplot here to check whether alcohol is normally distributed or is skewed.



- In our dataset we have a categorical features. With the countplot function, we are going to count the unique category in that feature. With for loop and subplot we have plotted this below graph.
- From the plot we came to know, count of white wine observations is much more than the red wine.



```
df_cat = data.select_dtypes(include='object')
df_cat.head()
```

	type
0	white
1	white
2	white
3	white
4	white

- The hist() function in pyplot module of matplotlib library is used to plot a histogram.

- As we can see, the most common vote is '6', when the lowest vote is '3', and the highest vote is '6'. In general, we may see that most of the parameters (except the "type" parameter, which is binary parameter) are normally distributed.

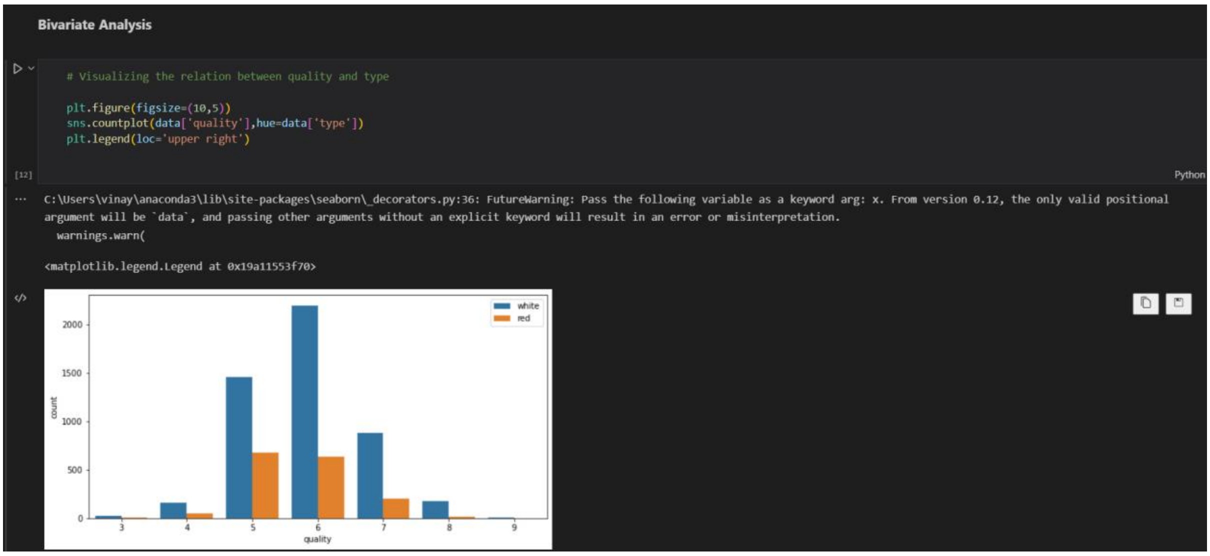
Bivariate Analysis

Activity 4: Bivariate analysis

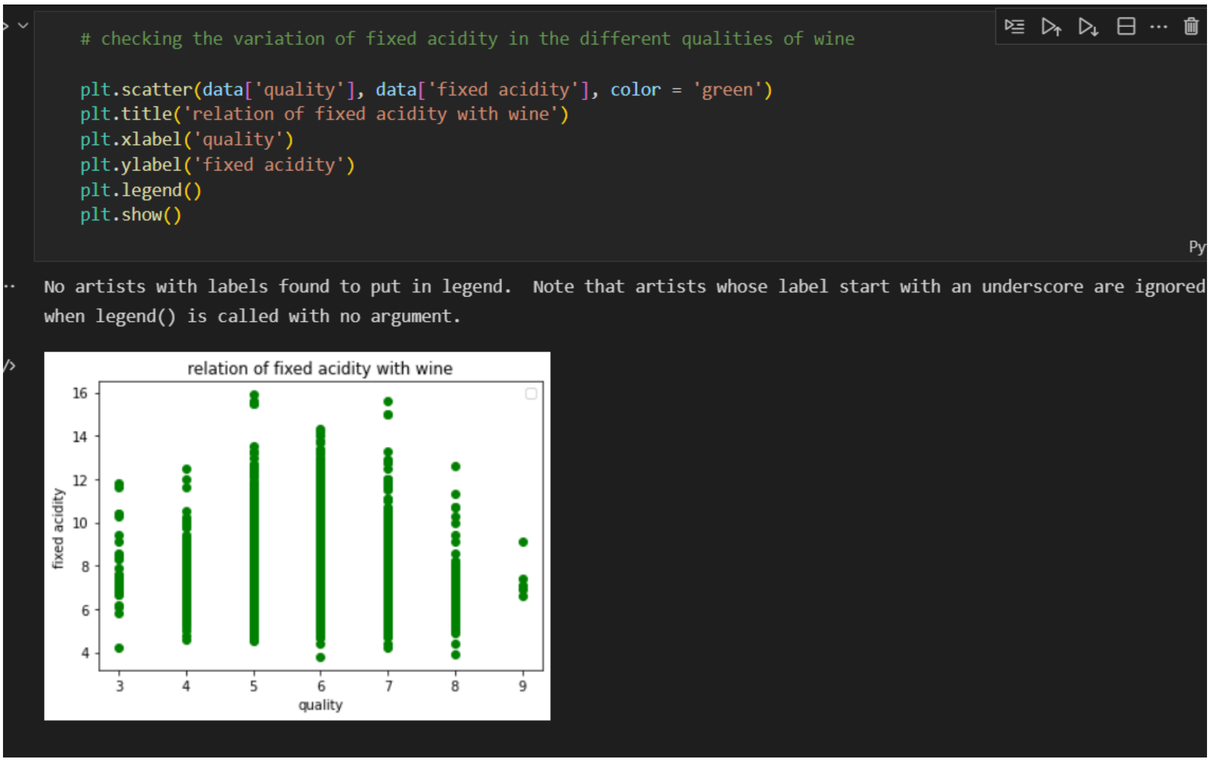
To find the relation between two features we use bivariate analysis.

- Countplot is used here. As a 1st parameter we are passing x value and as a 2nd parameter we are passing hue value.

- From this plot we can see the relationship between type and the quality of the data.



- A scatter plot is a means to represent data in a graphical format. A simple scatter plot makes use of the Coordinate axes to plot the points, based on their values. Scatter plots uses dots to represent individual pieces of data.
- The following scatter plot represents the relationship between quality and fixed acidity as a scatter plot.



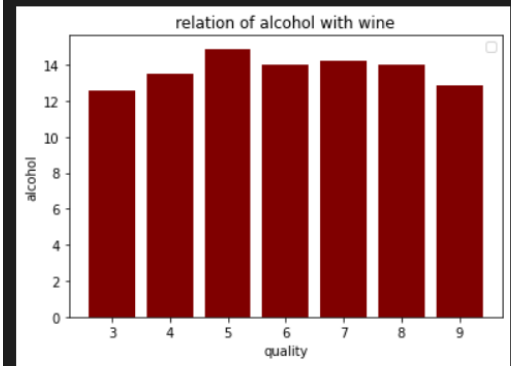
- A bar chart or bar graph is a chart or graph that presents categorical data with rectangular bars with heights or lengths proportional to the values that they represent.

- The following visualization represents the variation of fixed acidity in the different qualities of wine.

```python
# checking the variation of fixed acidity in the different qualities of wine

plt.bar(data['quality'], data['alcohol'], color = 'maroon')
plt.title('relation of alcohol with wine')
plt.xlabel('quality')
plt.ylabel('alcohol')
plt.legend()
plt.show()
```

Python

```
No artists with labels found to put in legend.  Note that artists whose label start with an underscore are ignored
when legend() is called with no argument.
```

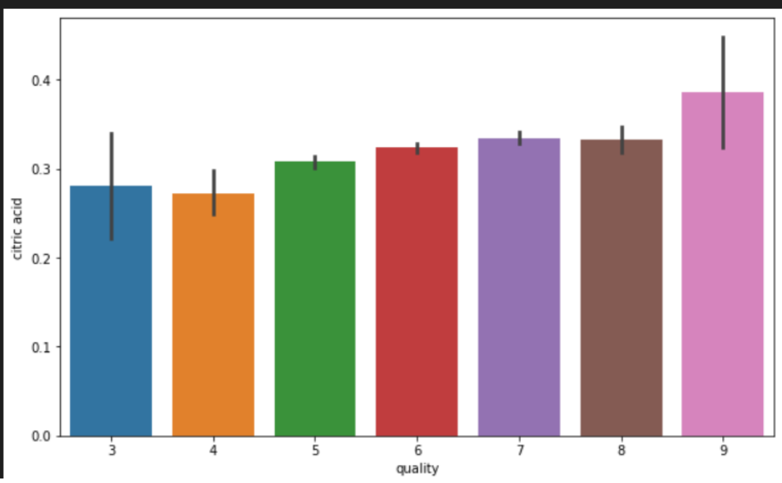


- Let’s see the relationship between the dependent and independent variables of our dataset.
- The following visualization represents the relationship between Citric Acid and our target variable, Quality. We can see there’s not much variation in citric acid values over the quality.

```python
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'citric acid', data = data)
```

Pytho

```
<AxesSubplot:xlabel='quality', ylabel='citric acid'>
```

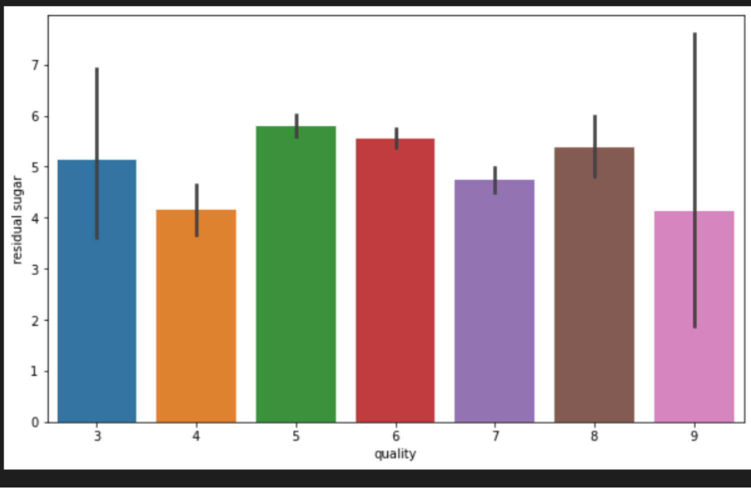


- The following visualization represents the relationship between Residual sugar and our target variable, Quality. We can see the variation in Residual sugar values over the quality.

```python
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'residual sugar', data = data)
```

Python

```
<AxesSubplot:xlabel='quality', ylabel='residual sugar'>
```

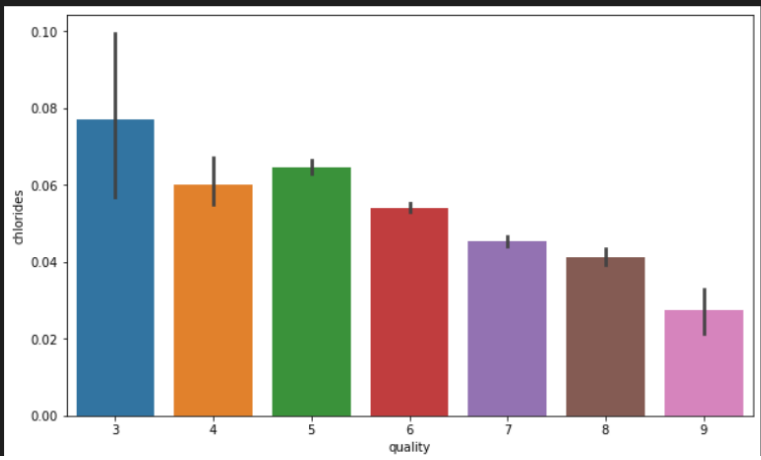


- The following visualization represents the relationship between Chlorides and our target variable, Quality. We can see the variation in Chlorides values over the quality.
- We can see that the Composition of chloride also go down as we go higher in the quality of the wine so, we can say that Chlorides and Quality are inversely related.

```python
#Composition of chloride also go down as we go higher in the quality of the wine

fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'chlorides', data = data)
```

Python

<AxesSubplot:xlabel='quality', ylabel='chlorides'>

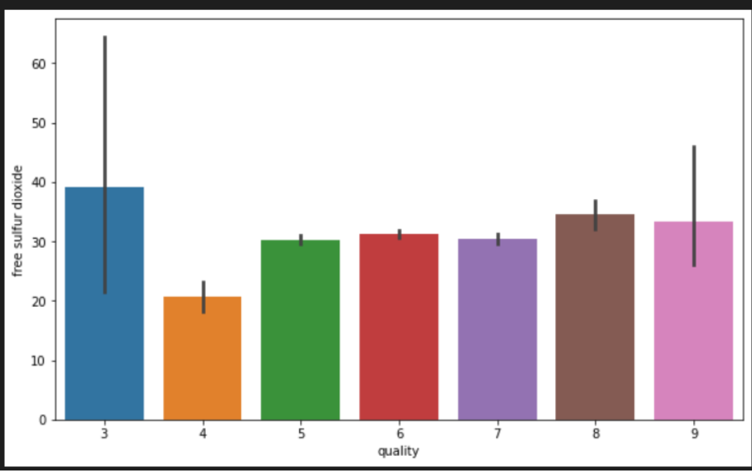


- The following visualization represents the relationship between free sulfur dioxide and our target variable, Quality. We can see the variation in free sulfur dioxide values over the quality.

```python
fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'free sulfur dioxide', data = data)
```

Python

<AxesSubplot:xlabel='quality', ylabel='free sulfur dioxide'>

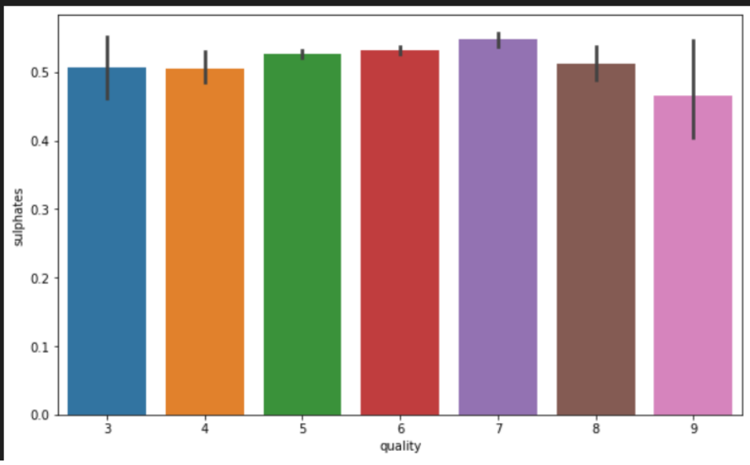


- The following visualization represents the relationship between sulphates and our target variable, Quality. We can see that there’s not much variation in sulphates values over the quality.

```python
#Sulphates level goes higher with the quality of wine

fig = plt.figure(figsize = (10,6))
sns.barplot(x = 'quality', y = 'sulphates', data = data)
```

Python

```
<AxesSubplot:xlabel='quality', ylabel='sulphates'>
```

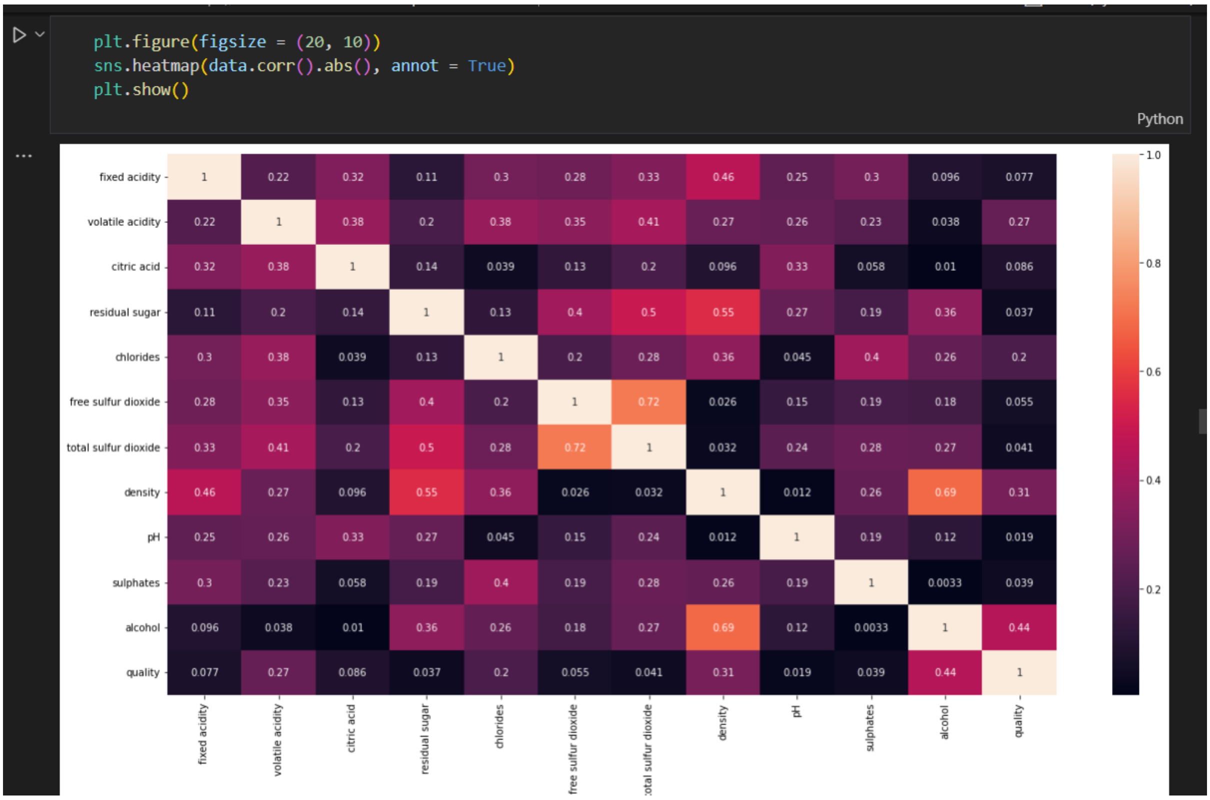


- As we can see that like the above two items do not have very strong relation to the dependent variable, we have to showcase a correlation plot to check which of the items are more related to the dependent variable and which items are less related to the dependent variables.

Multivariate Analysis

In simple words, multivariate analysis is to find the relation between multiple features. Here we have used a heatmap from the seaborn package.

- Correlation is a statistical measure that expresses the extent to which two variables are linearly related. It's a common tool for describing simple relationships without making a statement about cause and effect.
- To visualize the correlation heat map() function is used. From the below image we can easily find the highly correlated feature. Abs() method is used to convert the negative correlation to positive correlation.



- From the above correlation plot for the given dataset for wine quality prediction, we can easily see which items are related strongly with each other and which items are related weekly with each other. For Example,
- The strongly correlated items are :

  1.fixed acidity and citric acid. 2.free sulphur dioxide and total sulphor dioxide. 3.fixed acidity and density. 4. alcohol and quality.

  so, from above points there is a clear inference that alcohol is the most important characteristic to determine the quality of wine.

- The weakly correlated items are :

  1.citric acid and volatile acidity. 2.fixed acidity and ph. 3.density and alcohol.

  These are some relations which do not depend on each other at all.

Descriptive Analysis

Descriptive analysis is to study the basic features of data with the statistical process. Here pandas has a worthy function called describe. With this describe function we can understand the unique, top and frequent values of categorical features. And we can find mean, std, min, max and percentile values of continuous features.

```python
#Descriptive Analysis
data.describe()
```

Python

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH
count	6487.000000	6489.000000	6494.000000	6495.000000	6495.000000	6497.000000	6497.000000	6497.000000	6488.000000
mean	7.216579	0.339691	0.318722	5.444326	0.056042	30.525319	115.744574	0.994697	3.218395
std	1.296750	0.164649	0.145265	4.758125	0.035036	17.749400	56.521855	0.002999	0.160748
min	3.800000	0.080000	0.000000	0.600000	0.009000	1.000000	6.000000	0.987110	2.720000
25%	6.400000	0.230000	0.250000	1.800000	0.038000	17.000000	77.000000	0.992340	3.110000
50%	7.000000	0.290000	0.310000	3.000000	0.047000	29.000000	118.000000	0.994890	3.210000
75%	7.700000	0.400000	0.390000	8.100000	0.065000	41.000000	156.000000	0.996990	3.320000
max	15.900000	1.580000	1.660000	65.800000	0.611000	289.000000	440.000000	1.038980	4.010000

Data Pre-Processing

As we have understood how the data is. Lets pre-process the collected data.

The download data set is not suitable for training the machine learning model as it might have so much of randomness so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps.

- Removing unwanted columns
- Handling missing values
- Converting the target variable into binary class variable
- Handling categorical data
- Splitting dataset into training and test set
- Scaling Techniques

Note: These are the general steps of pre-processing the data before using it for machine learning. Depending on the condition of your dataset, you may or may not have to go through all these steps.

In the data frame, head() function is used to display the first 5 data. Our dataset has employee id (unique values), department (totally 9 dept.), region (location), education, gender, recruitment channel, age, no. of trainings, previous year ratings, length of service, KPIs, award won, average training score and is_promoted (target variable) columns.

```
data.head()
```

	type	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	white	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45	8.8	6
1	white	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	6
2	white	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6
3	white	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6
4	white	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6

Drop Unwanted Features

- drop() is used to drop specified labels from rows or columns.

  Remove rows or columns by specifying label names and corresponding axis, or by specifying directly index or column names.

- We are building the model to predict the Quality of wine. As we saw that volatile acidity, total sulfur dioxide, chlorides, density are very less related to the dependent variable quality so even if we remove these columns the accuracy won't be affected that much.

```python
data = data.drop(['volatile acidity', 'total sulfur dioxide', 'chlorides', 'density'], axis = 1)

# checking the shape of the dataset
print(data.shape)
```

Python

```
... (6497, 9)
```

Converting The Target Variable Into Binary Class Variable

We are predicting the quality of wine but our quality column has rating in the range 3-9, we need to change it into binary class i.e good or bad as the quality can be either good or bad.

```
# converting the response variables(3-7) as binary response variables that is either good or bad
data['quality'] = data['quality'].map({3 : 'bad', 4 :'bad', 5: 'bad',
                                       6: 'good', 7: 'good', 8: 'good'})
```

```
# analyzing the different values present in the dependent variable(quality column)
data['quality'].value_counts()
```

```
good    4108
bad     2384
Name: quality, dtype: int64
```

Value_counts() is used to count different values in the column.

Now, let’s see if our dataset has null values or not and if there’s any null value so we need to handle the null values as they can affect the accuracy.

Checking For Null Values

For checking the null values, df.isnull() function is used.

- To check if there is any null value or not. True indicates the respective column has null values and False indicates that there’s no null value in that column.

```
[29]  data.isnull().any()                                  Python

...  type                    False
     fixed acidity            True
     citric acid              True
     residual sugar           True
     free sulfur dioxide     False
     pH                       True
     sulphates                True
     alcohol                 False
     quality                  True
     dtype: bool
```

- To sum those null values we use .sum() function to it. From the below image we found that education column and previous year rating column has null values.

```
[30]  data.isnull().sum()                                  Python

...  type                    0
     fixed acidity          10
     citric acid             3
     residual sugar          2
     free sulfur dioxide     0
     pH                      9
     sulphates               4
     alcohol                 0
     quality                 5
     dtype: int64
                                   + Code   + Markdown
```

- Let’s handle the null values.
- For features like fixed acidity, suphates, pH, residual sugar, Citric acid we can replace the null values with their mean as these columns have numerical values. We can use mean() to calculate the mean

- For the Quality we can replace it with its respective mode[0] value.

```python
data["fixed acidity"].fillna(data["fixed acidity"].mean(),inplace = True)
data["sulphates"].fillna(data["sulphates"].mean(),inplace = True)
data["pH"].fillna(data["pH"].mean(),inplace = True)
data["residual sugar"].fillna(data["residual sugar"].mean(),inplace = True)
data["citric acid"].fillna(data["citric acid"].mean(),inplace = True)
data["quality"].fillna(data["quality"].mode()[0],inplace = True)
```
Python

Handling Categorical Values

As we can see our dataset has categorical data we must convert the categorical data to integer encoding or binary encoding.

To convert the categorical features into numerical features we use encoding techniques. There are several techniques but in our project we are using feature mapping and label encoding.

- In our project, categorical features are education and department feature. Feature mapping on education is done by replace() function.
- Label encoder is initialized and department feature is passed as parameter for fit_transform() function. Label encoding uses alphabetical ordering. In type feature we have 2 categories, white and red and in quality also we have 2 categories, good or bad. Those categories are labelled in alphabetical order.

```python
#converting categorical data to numerical data
le = LabelEncoder()
data['quality'] = le.fit_transform(data['quality'])
data['type'] = le.fit_transform(data['type'])
```
Python

Splitting Data Into Train And Test

Now let's split the Dataset into train and test sets. First split the dataset into x and y and then split the data set

```python
# dividing the dataset into dependent and independent variables
x = data.iloc[:,:8]
y = data.iloc[:,8:9]
# determining the shape of x and y.
print(x.shape)
print(y.shape)
```
Python

```
(6497, 8)
(6497, 1)
```

Now let's split the Dataset into train and test sets. For splitting training and testing data we are using train_test_split() function from sklearn. As parameters, we are passing x, y, test_size, random_state.

For deep understanding refer this link: https://www.geeksforgeeks.org/how-to-split-a-dataset-into-train-and-test-sets-using-python/

```python
# dividing the dataset in training and testing set
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.25, random_state = 44)

# determining the shapes of training and testing sets
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```
[36] Python

```
(4872, 8)
(4872, 1)
(1625, 8)
(1625, 1)
```

Feature Scaling

Feature scaling is a method used to normalize the range of independent variables or features of data. Standard scaler() is initialized. Independent training data is passed in the fit_transform() method and independent test data is passed in the transform() function.

```python
# standard scaling
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.fit_transform(x_test)
```
Python

Model Building

Now our data is cleaned and it’s time to build the model. We can train our data on different algorithms. For this project we are applying two regression algorithms. The best model is saved based on its performance.

Logistic Regression

Logistic Regression is a Machine Learning algorithm which is used for the classification problems, it is a predictive analysis algorithm and based on the concept of probability.

```python
def logisticRegression(x_train, x_test, y_train, y_test):
    # creating the model
    model = LogisticRegression()
    # feeding the training set into the model
    model.fit(x_train, y_train)
    # predicting the results for the test set
    y_pred = model.predict(x_test)
    # calculating the training and testing accuracies
    print('***logisticRegression***')
    print("Training accuracy :", model score: Any in, y_train))
    print("Testing accuracy :", model.score(x_test, y_test))
    # classification report
    print(classification_report(y_test, y_pred))
    # confusion matrix
    print(confusion_matrix(y_test, y_pred))
```
Python

Stochastic Gradient Descent Classifier Model

A function named SGD is created and train and test data are passed as the parameters. Inside the function, SGD Classifier algorithm is initialized and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, confusion matrix and classification report is done.

```python
def SGD(x_train, x_test, y_train, y_test):
    # creating the model
    model = SGDClassifier(penalty=None)
    # feeding the training model into the model
    model.fit(x_train, y_train)
    # predicting the values for the test set
    y_pred = model.predict(x_test)
    print('***Stochastic Gradient Descent Classifier***')
    print("Training accuracy :", model.score(x_train, y_train))
    print("Testing accuracy :", model.score(x_test, y_test))
    # classification report
    print(classification_report(y_test, y_pred))
    # confusion matrix
    print(confusion_matrix(y_test, y_pred))
```

Python

Support Vector Classifier Model

A function named SVClassifier is created and train and test data are passed as the parameters. Inside the function, SVC algorithm is initialized and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, confusion matrix and classification report is done.

```python
def SVClassifier(x_train, x_test, y_train, y_test):
    # creating the model
    model = SVC()
    # feeding the training set into the model
    model.fit(x_train, y_train)
    # predicting the results for the test set
    y_pred = model.predict(x_test)
    # calculating the training and testing accuracies
    print('***Support Vector Classifier***')
    print("Training accuracy :", model.score(x_train, y_train))
    print("Testing accuracy :", model.score(x_test, y_test))
    # classification report
    print(classification_report(y_test, y_pred))
    # confusion matrix
    print(confusion_matrix(y_test, y_pred))
```

Python

Decision Tree Model

A function named decisionTree is created and train and test data are passed as the parameters. Inside the function, DecisionTreeClassifier algorithm is initialized and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, confusion matrix and classification report is done.

```python
def decisionTree(x_train, x_test, y_train, y_test):
    dt=DecisionTreeClassifier()
    dt.fit(x_train,y_train)
    yPred = dt.predict(x_test)
    print('***DecisionTreeClassifier***')
    print("Training accuracy :", dt.score(x_train, y_train))
    print("Testing accuracy :", dt.score(x_test, y_test))
    print('Confusion matrix')
    print(confusion_matrix(y_test,yPred))
    print('Classification report')
    print(classification_report(y_test,yPred))
```
Python

Random Forest Regressor Model

A function named randomForest is created and train and test data are passed as the parameters. Inside the function, RandomForestClassifier algorithm is initialized and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, confusion matrix and classification report is done.

```python
def randomForest(x_train, x_test, y_train, y_test):
    rf = RandomForestClassifier()
    rf.fit(x_train,y_train)
    yPred = rf.predict(x_test)
    print('***RandomForestClassifier***')
    print("Training accuracy :", rf.score(x_train, y_train))
    print("Testing accuracy :", rf.score(x_test, y_test))
    print('Confusion matrix')
    print(confusion_matrix(y_test,yPred))
    print('Classification report')
    print(classification_report(y_test,yPred))
```
Python

XGBoost Model

A function named xgboost is created and train and test data are passed as the parameters. Inside the function, GradientBoostingClassifier algorithm is initialized and training data is passed to the model with .fit() function. Test data is predicted with .predict() function and saved in new variable. For evaluating the model, confusion matrix and classification report is done.

```python
def xgboost(x_train, x_test, y_train, y_test):
    xg = GradientBoostingClassifier()
    xg.fit(x_train,y_train)
    yPred = xg.predict(x_test)
    print('***GradientBoostingClassifier***')
    print("Training accuracy :", xg.score(x_train, y_train))
    print("Testing accuracy :", xg.score(x_test, y_test))
    print('Confusion matrix')
    print(confusion_matrix(y_test,yPred))
    print('Classification report')
    print(classification_report(y_test,yPred))
```

Now let's see the performance of all the models and save the best model

Compare The Models

For comparing the above four models compareModel function is defined.

```python
def compareModel(x_train, x_test, y_train, y_test):
    logisticRegression(x_train, x_test, y_train, y_test)
    print('-'*100)
    SGD(x_train, x_test, y_train, y_test)
    print('-'*100)
    SVClassifier(x_train, x_test, y_train, y_test)
    print('-'*100)
    decisionTree(x_train, x_test, y_train, y_test)
    print('-'*100)
    randomForest(x_train, x_test, y_train, y_test)
    print('-'*100)
    xgboost(x_train, x_test, y_train, y_test)
    print('-'*100)
```

```python
compareModel(x_train, x_test, y_train, y_test)
```

After calling the function, the results of models are displayed as output. From the six models, random forest is performing well. From the below image, we can see the train accuracy of the model is 81% accuracy. So, here random forest is selected as the best performing algorithm and evaluated with cross validation. Additionally, we can tune the model with hyper parameter tuning techniques.

```
***logisticRegression***
Training accuracy : 0.7066912972085386
Testing accuracy : 0.6904615384615385
              precision    recall  f1-score   support

           0       0.58      0.50      0.54       584
           1       0.74      0.80      0.77      1041

    accuracy                           0.69      1625
   macro avg       0.66      0.65      0.65      1625
weighted avg       0.68      0.69      0.69      1625

[[294 290]
 [213 828]]
```

```
----------------------------------------------------------------------------------------------------
***Stochastic Gradient Descent Classifier***
Training accuracy : 0.6863711001642037
Testing accuracy : 0.6695384615384615
              precision    recall  f1-score   support

           0       0.55      0.48      0.51       584
           1       0.73      0.78      0.75      1041

    accuracy                           0.67      1625
   macro avg       0.64      0.63      0.63      1625
...
   macro avg       0.68      0.69      0.68      1625
weighted avg       0.71      0.70      0.71      1625

----------------------------------------------------------------------------------------------------
```

```
***Support Vector Classifier***
Training accuracy : 0.7676518883415435
Testing accuracy : 0.7304615384615385
              precision    recall  f1-score   support

           0       0.65      0.55      0.59       584
           1       0.77      0.83      0.80      1041

    accuracy                           0.73      1625
   macro avg       0.71      0.69      0.70      1625
weighted avg       0.72      0.73      0.72      1625

[[321 263]
 [175 866]]
```

```
***DecisionTreeClassifier***
Training accuracy : 1.0
Testing accuracy : 0.7089230769230769
Confusion matrix
[[367 217]
 [256 785]]
Classification report
              precision    recall  f1-score   support

           0       0.59      0.63      0.61       584
           1       0.78      0.75      0.77      1041

    accuracy                           0.71      1625
   macro avg       0.69      0.69      0.69      1625
weighted avg       0.71      0.71      0.71      1625
```

```
***RandomForestClassifier***
Training accuracy : 1.0
Testing accuracy : 0.803076923076923
Confusion matrix
[[391 193]
 [127 914]]
Classification report
              precision    recall  f1-score   support

           0       0.75      0.67      0.71       584
           1       0.83      0.88      0.85      1041

    accuracy                           0.80      1625
   macro avg       0.79      0.77      0.78      1625
weighted avg       0.80      0.80      0.80      1625

----------------------------------------------------------------------------------------------------
```

```
***GradientBoostingClassifier***
Training accuracy : 0.791871921182266
Testing accuracy : 0.7390769230769231
Confusion matrix
[[347 237]
 [187 854]]
Classification report
              precision    recall  f1-score   support

           0       0.65      0.59      0.62       584
           1       0.78      0.82      0.80      1041

    accuracy                           0.74      1625
   macro avg       0.72      0.71      0.71      1625
weighted avg       0.73      0.74      0.74      1625
```

Evaluating Performance Of The Model And Saving The Model

- To understand cross validation, refer this link. https://towardsdatascience.com/cross-validation-explained-evaluating-estimator-performance-e51e5430ff85.

- Confusion Matrix: It is a matrix representation of the results of any binary testing

```
# Random Forest confusion matrix
confusion_matrix(y_test, y_pred)

array([[395, 189],
       [124, 917]], dtype=int64)

#Random Forest Cross validation score
model_eval = cross_val_score(estimator = rfmodel, X = x_train, y = y_train, cv = 5)
model_eval.mean()

0.8064432159216555
```

Let's now save the model

```
pickle.dump(rfmodel,open('wineQuality_new.pkl','wb'))
```

Application Building

This section will build a web application integrated into the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.
This section has the following tasks

- Building HTML Pages
- Building server-side script

Building HTML Pages

- We will create the front-end part of the web page on this HTML page. On this page, we will accept input from the user and Predict the values.
  For more information regarding HTML
  https://www.w3schools.com/html/
  In our project we have HTML files, they are
  1. home.html

2. index.html
3. pred.html

home.html

```html
<!DOCTYPE html>
<html>
<head>
    <title>Wine Quality Prediction Using Machine Learning</title>
    <!-- Latest compiled and minified CSS -->
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

    <style type="text/css">
body {
  background-image: url('https://w0.peakpx.com/wallpaper/880/804/HD-wallpaper-good-wine-red-glass-nice-drop-wine-good-beautiful-abstract.jpg')
    </style>
</head>
<body>

<nav class="navbar">
  <div class="container-fluid">
    <div class="navbar-header">
      <a class="navbar-brand" style="color:white;font-size:250%;">Wine Quality Prediction</a>
    </div>
    <ul class="nav navbar-nav navbar-right">
        <li><a href="#" style="color:white;font-size:250%;">Home</a></li>
      <li><a href="/predict" style="color:white;font-size:250%;">Predict</a></li>
    </ul>
  </div>
</nav>

<div class="container">
  <h3 style="font-size:200%;color:#F96E6E;">Introduction</h3>
  <p style="color:white;font-size:120%;">Wine is the most healthful and most hygienic of beverages.
  Yes, if you think deep down then you just notice that we are discussing wine, above quote seems to be
   right because all over the world wine was soo popular among people, and 5% of the population doesn’t
   know what is wine? sounds good.<br><br>
    We definitely came across the fruit graphs, which is soo sweet on the test but graphs are not just to
     eat, they are used to make different types of things. Wine is one of them Wine is an alcoholic drink
     that is made up of fermented grapes. If you have come across wine then you will notice that wine has
     also their type they are red and white wine this was because of different varieties of graphs.<br><br>
     You are shocked to hear that the worldwide distribution of wine is 31 million tonnes which were huge
      in number. </p>

    <h3 style="font-size:150%;color:#F96E6E;">What if you think about the quality of wine, how can you
    differentiate the wine according to their quality? The big question arises.</h3>
     <p style="color:white;font-size:120%;"> According to experts, the wine is differentiated
      according to its smell, flavor, and color, but we are not a wine expert to say that wine is
       good or bad. What will we do then? Here’s the use of Machine Learning comes, yes you are thinking
        to write we are using machine learning to check wine quality.</p>

</div>


</body>
</html>
```

The HTML page looks like

It will display all the input parameters and the prediction text will display the output value of the data given by the user.

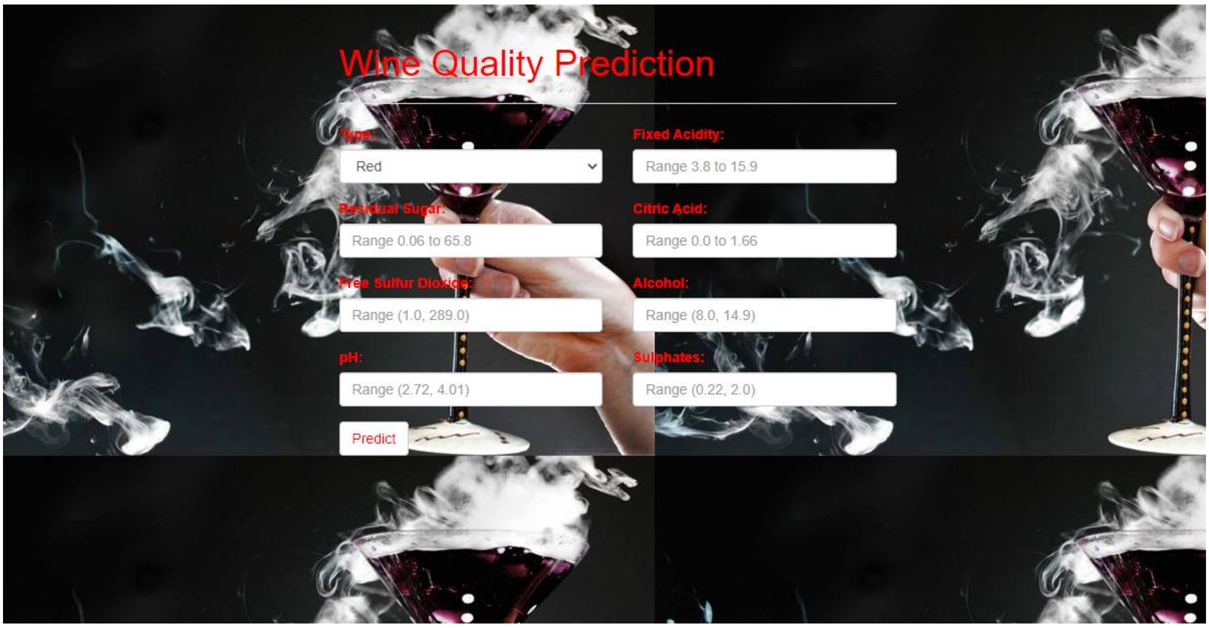
Index.html

```html
<!DOCTYPE html>
<html>
<head>
    <title>GDP Analysis</title>
    <!-- Latest compiled and minified CSS -->
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

    <style type="text/css">

body {
  background-image: url('https://res.allmacwallpaper.com/get/macbook-air-wallpapers/Name-Your-Poison/7799-720.jpg');
}

    </style>
</head>
<body>

    <div class="container">
        <div class="row">
            <div class="col-md-3"></div>
            <div class="col-md-6">
                 <div class="page-header">
                    <h1 style="color:red;">Wine Quality Prediction</h1>
                </div>
            </div>
        </div>
    </div>

    <div class="container">
        <div class="row">
            <div class="col-md-3"></div>
            <div class="col-md-6">
                <form action="/data_predict" method="POST">
                    <div class="row">
```

```html
                    <div class="row">
                        <div class="col-md-6">
                            <div class="form-group">
                                <label for="type" style="color:red;">Type:</label>
                                <select name="type" class="form-control" id="type">
                                    <option value="0">Red</option>
                                    <option value="1">White</option>
                                </select>
                            </div>
                        </div>

                        <div class="col-md-6">
                            <div class="form-group">
                                <label for="fixed_acidity" style="color:red;">Fixed Acidity:</label>
                                <input type="text" class="form-control" id="fixed_acidity" name="fixed_acidity" placeholder="Range 3.8 to
                            </div>
                        </div>
                    </div>

                    <div class="row">
                        <div class="col-md-6">
                            <div class="form-group">
                                <label for="residual_sugar" style="color:red;">Residual Sugar:</label>
                                <input type="text" class="form-control" id="residual_sugar" name="residual_sugar" placeholder="Range 0.06
                            </div>
                        </div>
                        <div class="col-md-6">
                            <div class="form-group">
                                <label for="citric_acid" style="color:red;">Citric Acid:</label>
                                <input type="text" class="form-control" id="citric_acid" name="citric_acid" placeholder="Range 0.0 to 1.6
                            </div>
                        </div>
                    </div>
```

```html
                        <div class="col-md-6">
                            <div class="form-group">
                                <label for="alcohol" style="color:red;">Alcohol:</label>
                                <input type="text" class="form-control" id="alcohol" name="alcohol" placeholder="Range (8.0, 14.9)">
                            </div>
                        </div>
                        </div>
                    <div class="row">
                        <div class="col-md-6">
                            <div class="form-group">
                                <label for="pH" style="color:red;">pH:</label>
                                <input type="text" class="form-control" id="pH" name="pH" placeholder="Range (2.72, 4.01)">
                            </div>
                        </div>

                        <div class="col-md-6">
                            <div class="form-group">
                                <label for="sulphates" style="color:red;">Sulphates:</label>
                                <input type="text" class="form-control" id="sulphates" name="sulphates" placeholder="Range (0.22, 2.0)">
                            </div>
                        </div>
                        </div>

                    <button type="submit" class="btn btn-default" style="color:red;">Predict</button>
                </form>
            </div>
        </div>
    </div>


    <!-- Latest compiled and minified JavaScript -->
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script
</body>
</html>
```

The HTML page looks like



Pred.html

```html
<!DOCTYPE html>
<html>
<head>
    <title>Wine Quality Prediction Using Machine Learning</title>
    <!-- Latest compiled and minified CSS -->
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

    <style type="text/css">
body {
  background-image: url('https://m.economictimes.com/thumb/msid-70001605,width-1200,height-900,resizemode-4,imgsize-541593/pretty-wine-bot
}
    </style>
</head>
<body>
    <div class="container">
        <div class="row">
            <div class="col-md-3"></div>
            <div class="col-md-6">
                 <div class="page-header">
                    <h1 style="font-size:500%;color:red;">Wine Quality Prediction</h1>
                </div>
            </div>
        </div>
    </div>
    <div class="container">
        <div class="row">
            <div class="col-md-3"></div>
            <div class="col-md-6">
                <div class="p-2 my-2 border">
                    <h3 style="font-size:300%;color:red;">Wine Quality is predicted to be :- {{prediction}}</h3>
                </div>
            </div>
        </div>
    </div>
```

Building Python Code

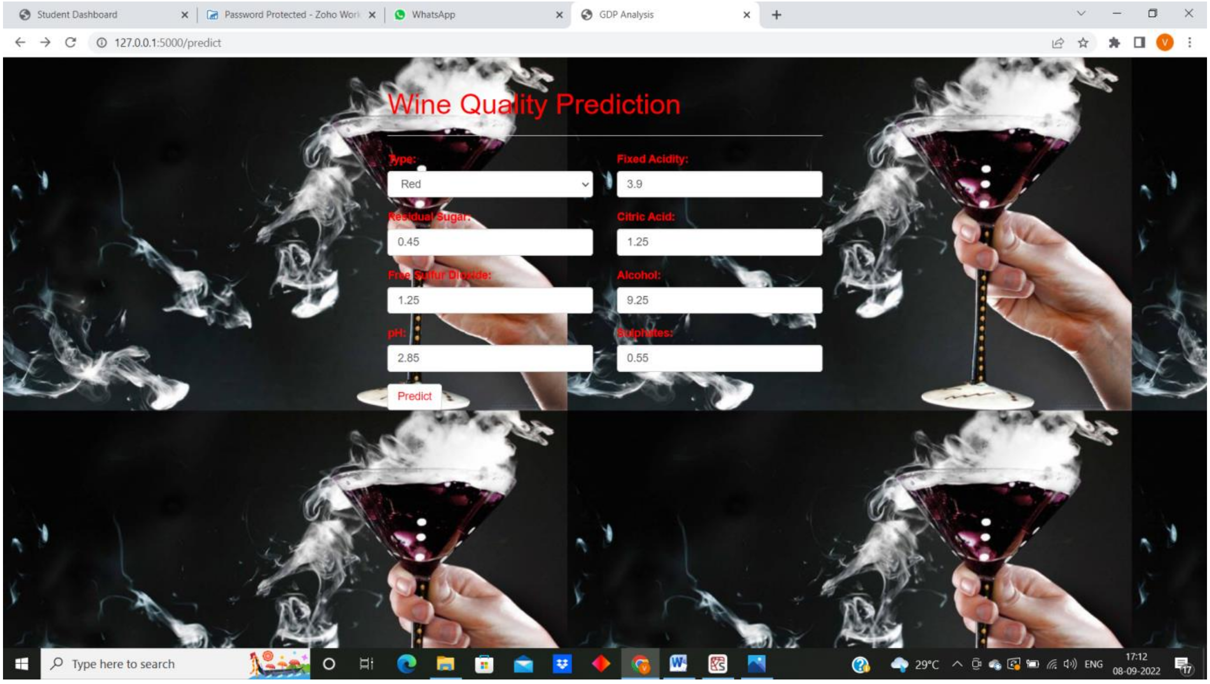
Let us build an app.py flask file which is a web framework written in python for server-side scripting. Let’s see the step by step procedure for building the backend application.

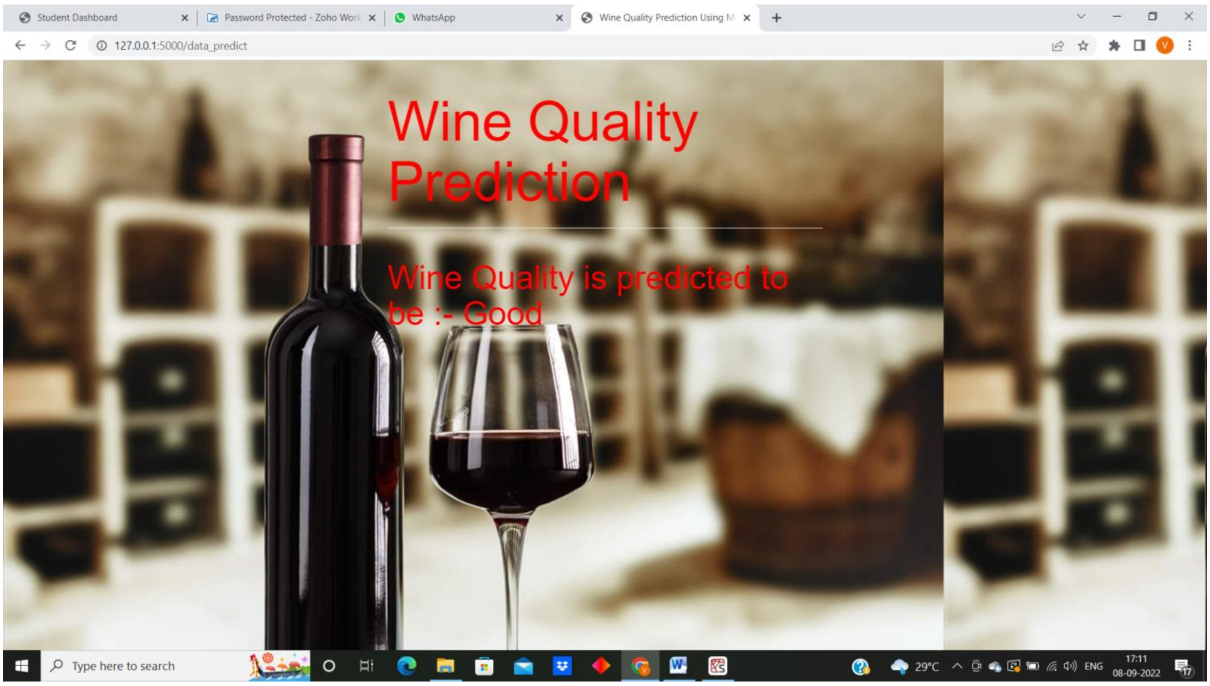
```python
from flask import Flask, render_template, request # Flask is a application
# used to run/serve our application
# request is used to access the file which is uploaded by the user in out application
# render_template is used for rendering the html pages
import pickle # pickle is used for serializing and de-serializing Python object structures


app=Flask(__name__) # our flask app

@app.route('/') # rendering the html template
def home():
    return render_template('home.html')
@app.route('/predict') # rendering the html template
def index() :
    return render_template("index.html")

@app.route('/data_predict', methods=['GET','POST']) # route for our prediction
def predict():

    # loading model which we saved
    model = pickle.load(open('wineQuality_new.pkl', 'rb'))

    data = [[x for x in request.form.values()]]

    pred= model.predict(data)[0]
    print(pred)
    if pred==0:
        prediction="Bad"
    else:
        prediction="Good"

    return render_template('pred.html', prediction=prediction)

if __name__ == '__main__':
    app.run(debug=False)
```

Output

- Open the new google tab and enter localhost:5000,it will display the form page
- Enter the values as per the form and click on predict button
- It will redirect to the page based on prediction output

S


Student Dashboard
Password Protected - Zoho Work
WhatsApp
GDP Analysis
127.0.0.1:5000/predict
Wine Quality Prediction
Type:
Red
Fixed Acidity:
3.9
Citric Acid:
0.45
1.25
Alcohol:
1.25
9.25
pH:
2.85
0.55
Predict
Type here to search
29°C
ENG
17:12
08-09-2022


Student Dashboard
Password Protected - Zoho Work
WhatsApp
127.0.0.1:5000/data_predict
Wine Quality Prediction
Wine Quality is predicted to be :- Good
Type here to search
29°C
ENG
17:11
08-09-2022